A Technical Report on WebGIS Application for House Discovery

1st

Shengyi Zhang *1097063*

Lakehead University

zhangs@lakeheadu.ca

2nd

Beili Yin *1148875*

Lakehead University

byin@lakeheadu.ca

CONTENTS

List of Figures

List of Tables

A Technical Report on WebGIS Application for House Discovery

***Abstract*—This article is the technical report of our Research Methodology course final project - WebGIS Application for House Discovery, a web application for house query and analysis. The report includes introduction, background, methodology (overall settings, interface design, data structure design, database design, interface design, etc.), results, discussion, conclusions and future works.**

***Keywords*—House Discovery, GIS, WebGIS, Springboot, Spring Security, Hibernate, Web Spider**

I. Introduction

With the rapid development of the Internet and mobile applications, GIS has gone from desktop map editing software operated by professionals to WebGIS applications that can be seen everywhere today. For example, map navigation software has become an indispensable application in our daily lives. Usually what we view and connect is discrete data, which does not contain location information. The data processed by Geographic Information System is the data that generally contains location information such as latitude and longitude. GIS data formats can be divided into vector data and raster data. Vector data is Generated by points, lines and surfaces, can be enlarged without loss, while raster data is usually composed of common sliced pictures such as png format.

This project aims to develop an application for finding a house information through a map, combining relational data and GIS data to provide users with a multi-angle house selection plan. The report mainly solves and realizes the analysis of requirements and the module design according to the requirements, including how to divide the system into several modules, explaining the interface between each module, the messages transmitted between modules, and the design of database structure. This report will give a detailed description of the design of this system.

II. Problem Definition

Traditionally, people use paper media and intermediaries to publish and view housing information, which is relatively inefficient. Nowadays, people generally search and view housing through websites and online maps. For example, the two largest online house information websites in China and the United States, such as trulia and beike. Real estate trading platform, but after several trials and researches, it is found that such platforms cannot meet the requirement of finding houses by viewing nearby facilities and buildings, so we decided to develop a bus, hospital, school, shopping mall, office buildings, etc. within a certain distance through the annex to filter out housing information retrieval applications. The overall process is to use spring to develop a crawler that regularly crawls real estate website data every day. Here, we take the listings in Lianjia.com-Shanghai area as an example. The data is scrapped from Lianjia and then stored in the database. In addition, we used Java Springboot, hibernate, Spring Security for software development . The permission controller end is used to process front-end requests and return restful API. The front end selects the LayUI framework for web page development. The front-end and back-end of our project were developed in separated Model-View-Controller-Service architecture.

This application can count and display key information such as the number of houses available for sale in all districts in Shanghai, the average price per square meter, the average total price and other key information. Then the user can click on any district, select the query distance, and search conditions, on the map Mark the buses, hospitals, schools, shopping malls, and office buildings attached to the community, and update the query results to the database simultaneously.

Based on the collected information and the different data tables established, the house analysis function is realized, and the radar chart is used to present it.

Based on the collected housing information with geographic information, the distribution of housing prices in different areas is displayed on the map in the form of a heat map.

III. Preliminary preparation

Software: IDEA is used to write java code, MySQL, Redis is used for database storage, Navicat is used for database viewing, and VScode is used for front-end development (HTML, CSS, JavaScript)

Hardware: Cloud servers are used for project deployment and front-end and back-end communication testing.

IV. Requirements Analysis

In this section, we will go through all the requirements of each module of the WebGIS Application from data collection to house information filtering based on price.

A. Data Collection

Introduction: Regularly obtain and update the details of the houses for sale in all the streets and all the communities in all areas of Shanghai.

Input: city, task execution time.

Output: None.

B. User Login System

Introduction: Basic user login, you can enter the registered mailbox and user name to log in.

Input: username or email.

Process:

1. The user enters the user name or email address and password.

2. LayUI checks the length of the password and the format of the email. If it succeeds, it will proceed to the next step. If it fails, it will pop up a layer prompt.

3. The front end sends a request request, and the username/email and password packaged in json format are used in the post body.

4. The backend receives the request to perform the authentication process of spring security. It first passes the verification of the user name and password, and then the verification of the authority role. If it is successful, it returns the jwt token and the status code of code: 0, msg: success.

5. After the front end receives the response, it stores the jwt token in the browser's localstorage so that the token will be carried in each subsequent request header.

6. The page jumps, a pop-up window of successful login pops up, and then the login page jumps to the home page.

C. Map Display

Introduction: As the gis data is displayed according to the latitude and longitude, it is necessary to obtain the longitude and latitude information of the house, and then query the building information near the house through the longitude and latitude of the house, and update the query result to the database

Input: Query the type and distance of nearby buildings, the unit is m

Process:

1. When the user enters the homepage, the left side displays all community information in a table, including city, region, street, community, average total price, average unit price, and available houses; the right side displays a map of Shanghai.

2. The user clicks on a row of the form, a pop-up form selection, showing the currently selected street and community name, the user can enter the building type and range to be retrieved, such as bus, 1000m, which means to retrieve 1000 bus stops around the community.

3. The front end requests the user to extract the requested parameters, and then calls the map api to query the latitude and longitude information of the neighborhood and street. If the query is successful, mark the neighborhood on the map on the right, and then call the map api's api for querying nearby building information according to the latitude and longitude

to obtain the basic information of the surrounding specific buildings, including the name, latitude and longitude, and the distance from the neighborhood, at the same time The map draws a circle with the specified query range, marks the buildings in the range, and packs the community coordinates and surrounding building information into json format.

4. Put the data packaged in a specific json format into the request body, then use ajax to send the request to the corresponding background api, and add the saved jwt token to the request header.

5. After receiving the request in the background, it analyzes the parameters and uses DTO to extract the request parameters.

6. First check whether there are latitude and longitude of the houses in the community through the name of the community in the request parameters. If so, do not update, if not, update the latitude and longitude of all houses in the community.

7. Query whether the building has been saved in the community through the array corresponding to each building retrieved in the request parameters. If it is saved, it will not be updated. If it is not saved, the nearby building information of all houses in the community will be updated.

D. Information Update

Introduction: Ordinary users can search and view, and administrators can update the information of nearby buildings

Input: form to select the building to be updated

Process:

1. The administrator selects the type of building that needs to be updated through the input form.

2. Click Submit.

3. The front end listens to the submission event, and then calls the background api to get a list of all the cells.

4. Since the front-end is operated asynchronously, in order to prevent frequent requests, a timer is set, and a cell is taken from the cell list every 1s for query.

5. According to the name of the community, the program calls the map api to query the latitude and longitude information of the community and street. If the query is successful, mark the neighborhood on the map on the right, and then call the map api's api for querying nearby building information according to the latitude and longitude to obtain the basic information of the surrounding specific buildings, including the name, latitude and longitude, and the distance from the neighborhood, at the same time The map draws a circle with the specified query range, marks the buildings in the range, and packs the community coordinates and surrounding building information into json format.

6. Put the data packaged in a specific json format into the request body, then use ajax to send the request to the corresponding background api, and add the saved jwt token to the request header.

7. After receiving the request in the background, it analyzes the parameters and uses DTO to extract the request parameters.

8. First check whether there are latitude and longitude of the houses in the community through the name of the community in the request parameters. If so, do not update, if not, update the latitude and longitude of all houses in the community.

9. Query whether the building has been saved in the community through the array corresponding to each building retrieved in the request parameters. If it is saved, it will not be updated. If it is not saved, the nearby building information of all houses in the community will be updated. 10. If all cells in the cell list have been traversed, clear the timer and end the loop.

E. House Price Distribution

Introduction: Users can click on the heat map on the left menu to display the high and low distribution of housing prices in different areas of Shanghai.

Input: The user clicks the menu on the left, and the main body on the right renders the corresponding page.

Process:

1. Monitor the user's click event, and then load the page.

2. The front-end sends a request to the back-end to obtain all housing geographic information and housing price data.

3. The background receives the request, queries all housing geographic information and housing price data, and returns housing information with latitude, longitude and housing price.

4. The front end receives the request and calls echarts and Baidu map api on the page to draw the heat map.

F. *Screening Evaluation*

Introduction: Users can click on the house evaluation menu on the left menu to conduct house information evaluation. The evaluation is presented as a radar chart.

Input: The user selects the cell on the left, and the radar chart.

Process:

1. The user selects the community to be evaluated through the form on the upper left side, and clicks submit.

2. The front-end obtains the requested cell name, packs it into json format, and sends the request to the back-end.

3. After receiving the name of the community in the background, first retrieve all the houses available for sale in the community from the database, and then query the surrounding building information of the community through service, and put the different building information into the corresponding array, which is stored in the key-value type, and the key is Corresponding to the name of the building, value is the array obtained by the retrieval, and then the response is packaged into the front end format, and all data is placed in data. If there is no relevant information, the value of the corresponding key returns an empty array.

4. The front end receives the request and renders the form according to the response body. First, extract all the housing information in the community, and display the saleable houses in the community at the bottom left. If the houses are an empty array, only the header is rendered.

5. At the same time, use echarts to draw the radar map with the information of buildings near the cell extracted from the response body, and display the information of nearby buildings with icons.

V. Conditions and Restrictions

Time limit: The data scrapped by web spiders is limited, and housing information is hidden due to the regulation of house prices.

Historical data: historical data is missing, invalid for effective machine learning and deep learning training to predict housing prices.

Law: Since the data is scrapped from a third-party website, all this application is for learning and communication purposes only, not for commercial use.

Server: The amount of real estate data in various cities in China is too large, especially when historical data and building information near houses need to be stored, and the server is under pressure when updating. The method adopted here is to use the timer to update the front-end and only Save the data of the latest day, and only save the housing price information in Shanghai.

VI. System Design

A. *Overall Pipeline*

As shown in Fig 1.

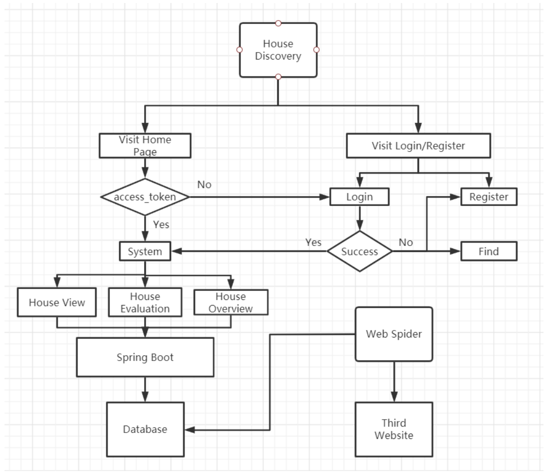


Fig. 1. Overall Pipeline

B. *Basic Client Pipeline*

The Client Front provides a cloud platform in the browser, establish communication with the background, monitor user events, generate and send corresponding requests and actions, receive the response from the server, and then render the page. As shown in Fig 5.

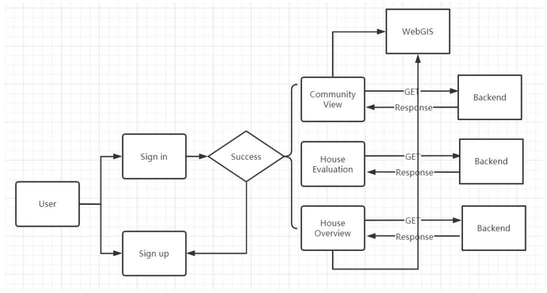


Fig. 2. Basic Client Pipeline

C. Basic Server Pipeline

The Server is designed to receive actions and requests sent by the client, process and respond, and at the same time respond to the operation and processing of the database. As shown in Fig 3.

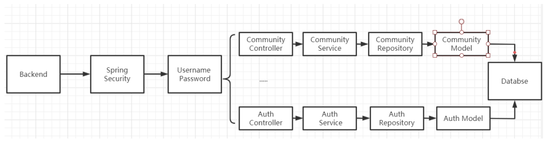


Fig. 3. Basic Server Pipeline

D. Web Spider Pipeline

The Web Spider can first analyze the web page structure, confirm the xml path of the data that needs to be scrapped, and then use the crawler framework webmagic to continuously send requests to get the response pages, extract the specified elements of the page, and then store the extracted data in the database. As shown in Fig 4.

E. Structure Design

Users are divided into ordinary users and administrator users. Ordinary users log in to the system with the role, and then retrieve and evaluate housing information; administrator users log in to the system with the role of admin, and can update the building information in the neighborhood.

Ordinary users log in to the system with their username or email, and after passing the authentication in the background, the json web token is returned, and then the user can view the

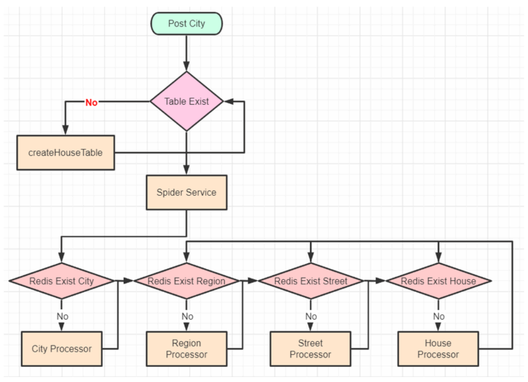


Fig. 4. Web Spider Pipeline

information of all the houses for sale in the community through the form, and click on a row of the form to further filter the nearby building information and display the map. , You can set the name and distance of the nearby buildings that need to be retrieved. The front-end first retrieves and draws through the map API, and then sends a request to the back-end to see if the information associated with the cell needs to be updated.

The administrator can uniformly update the building information near the cell, select the building type, send a request to the front end to obtain a list containing all cell information, and then traverse the list regularly, obtain the latitude and longitude and nearby building information through the map API, and then send the result Store and update data in the background.

F. Client Module

This Module contains user login and registration modules, user role registration, and the administrator is currently operated and set by the database administrator. As shown in Fig 5.

- Homepage: Residential housing price and geographic information page, including data form module and map module.
- House analysis: house analysis page, including data form module and heat map and radar map module drawn by echarts.
- User information: user information module
- System management module: user, role, menu management module.

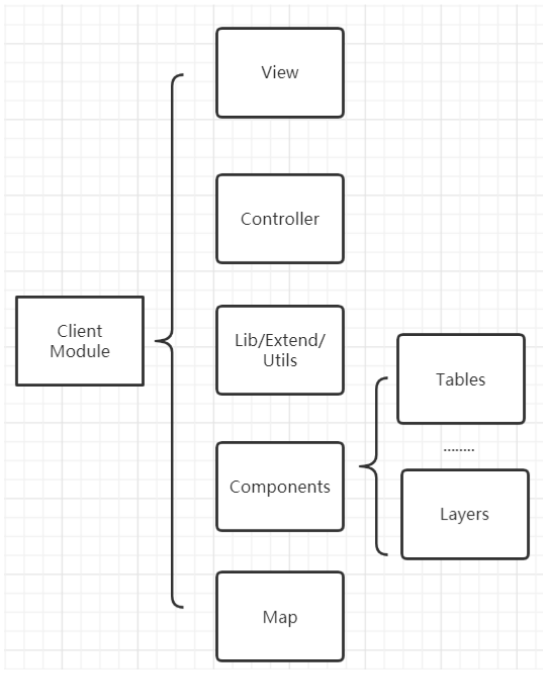


Fig. 5. Client Module

G. Server Module

The server mainly includes user registration, login, spring security authentication and authorization modules, as well as modules corresponding to different models, such as model, dto, controller, service, repository, etc. corresponding to house. Different models involve different queries and database modifications. As shown in Fig 6

H. Web Spider Module

The web spider module contains different processors to extract information such as city, house, region, street, etc., as well as modules to start crawlers regularly, and various tool modules.

VII. API Design

All application interfaces start with api, followed by the version number, such as api/v1. Due to the limited time this time, there is no distinction between developing, testing and release versions, and swagger is not used for interface management. Normal development should set different APIs

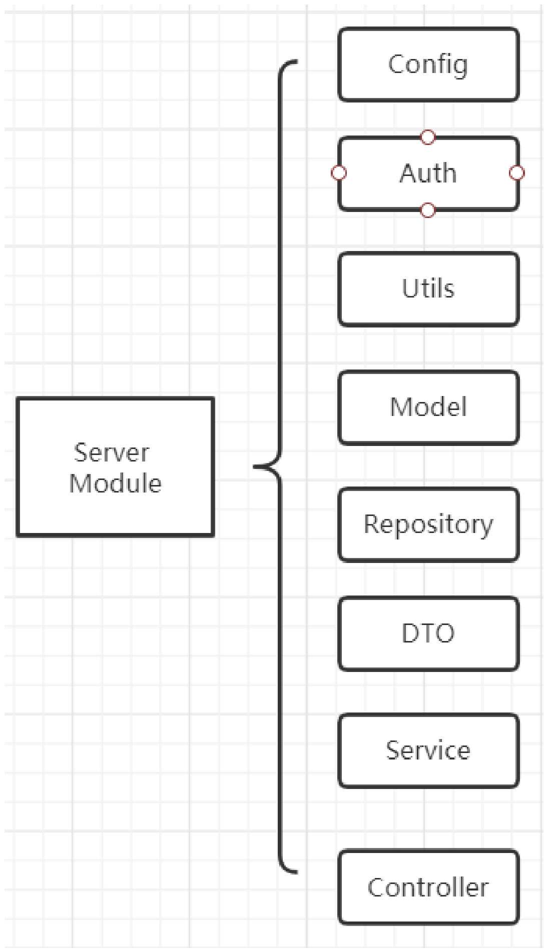


Fig. 6. Server Module

for development, testing and release, as well as database configuration, etc.

A. User API

- Login:
  Type: POST
  Link: /api/auth/signin
- Registration:
  Type: POST
  Link: api/auth/signup
- Acquire User information:
  Type: GET
  Link: /api/user/profile

B. *House Information API*

- Get all house information:
  Type: GET
  Link: /api/v1/houses
- Get all community information:
  Type: GET
  Link: /api/v1/community
- Update a certain community information:
  Type: POST
  Link: /api/v1/community

C. *Internal API*

- Get second-hand houses of Lianjia in the specified city:
  Type: GET
  Link: /house/city to crawl
- Get the second-hand houses of Lianjia in batch cities:
  Type: GET
  Link: /house/citys
- Get the second-hand houses of Lianjia nationwide:
  Type: GET
  Link: /house/nation
- Get the second-hand houses of chain houses across the country (exclude the designated cities):
  Type: GET
  Link: /house/nation/exclude

VIII. Data Structure

This section will demonstrate database environment, data naming rules and logical design.

A. *Database Environment Description*

Based on MySQL database system, the project relies on utf8mb4 for encoding, and set encoding=utf8 for database connection.

B. *Database Naming Rules*

We use the name as the table name, and add ”-” for the intermediate table to connect the two tables.

C. *Logical Design*

Users and Roles are a many-to-many relationship. As shown in Fig 7.

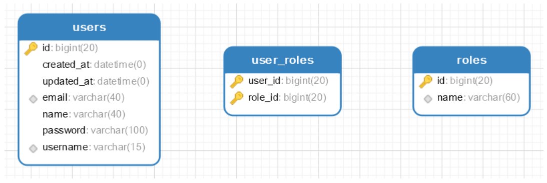


Fig. 7. User-Role Relationship

House has a many-to-many relationship with data that has information about bus, shop, school, hospital, etc. As shown in Fig **??**, 10, **??** and 11.

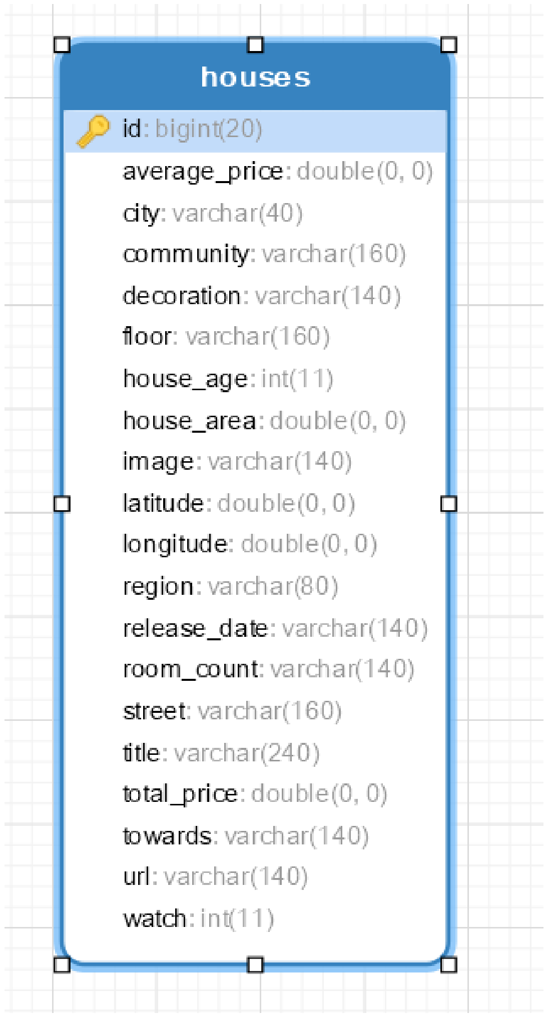


Fig. 8. House Data Structure

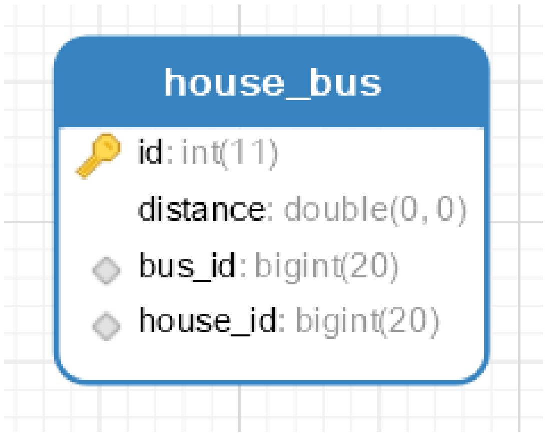


Fig. 9. House-Bus Relationship

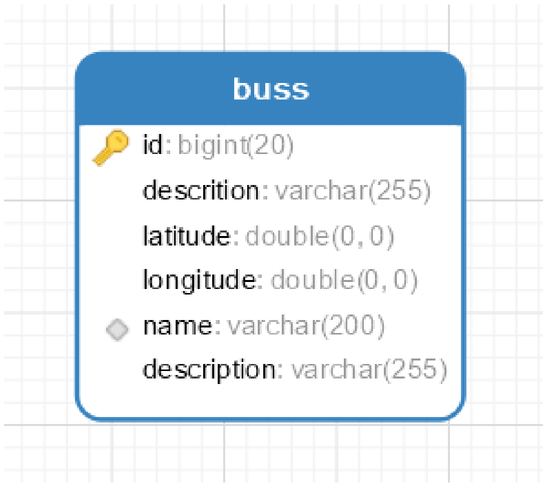


Fig. 10. Bus Data Structure

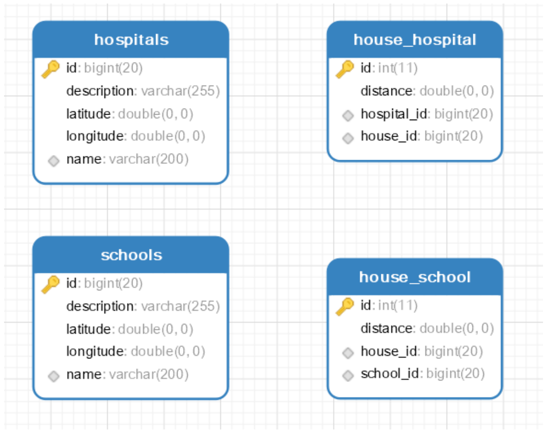


Fig. 11. Hospital-School Relationship

IX. UI Design

A. User Login

The login page has As shown in Fig 12

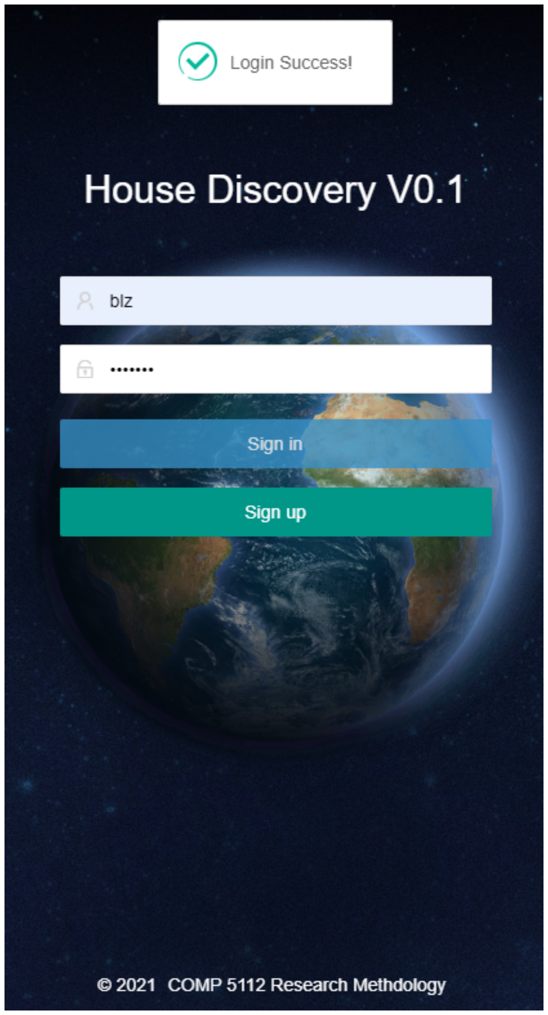


Fig. 12. House Discovery login

B. House Information Query

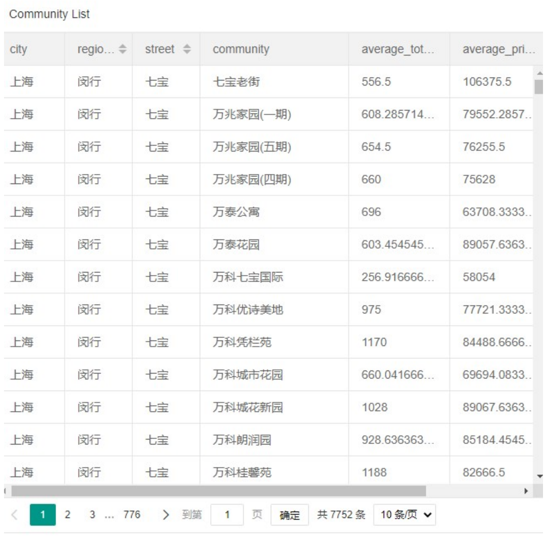
As shown in Fig 13

C. Interactive Map

As shown in Fig 14

D. Heat Map

As shown in Fig 15



Community List

city	regio...	street	community	average_tot...	average_pri...
上海	闵行	七宝	七宝老街	556.5	106375.5
上海	闵行	七宝	万兆家园(一期)	608.285714...	79552.2857...
上海	闵行	七宝	万兆家园(五期)	654.5	76255.5
上海	闵行	七宝	万兆家园(四期)	660	75628
上海	闵行	七宝	万泰公寓	696	63708.3333...
上海	闵行	七宝	万泰花园	603.454545...	89057.6363...
上海	闵行	七宝	万科七宝国际	256.916666...	58054
上海	闵行	七宝	万科优诗美地	975	77721.3333...
上海	闵行	七宝	万科凭栏苑	1170	84488.6666...
上海	闵行	七宝	万科城市花园	660.041666...	69694.0833...
上海	闵行	七宝	万科城花新园	1028	89067.6363...
上海	闵行	七宝	万科朗润园	928.636363...	85184.4545...
上海	闵行	七宝	万科桂馨苑	1188	82666.5

< 1 2 3 ... 776 > 到第 1 页 确定 共 7752 条 10 条/页

Fig. 13. House Information Query

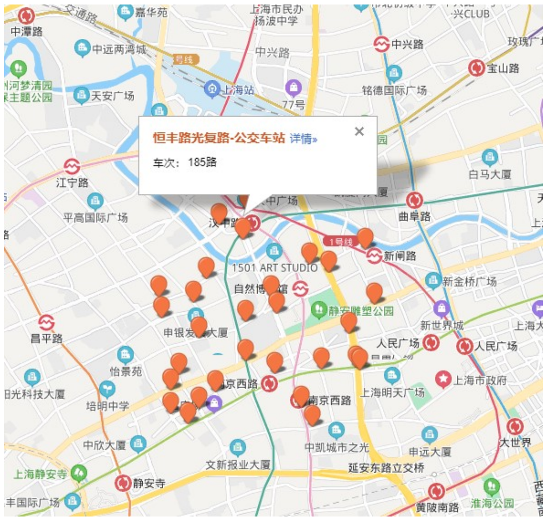


Fig. 14. Interactive Map

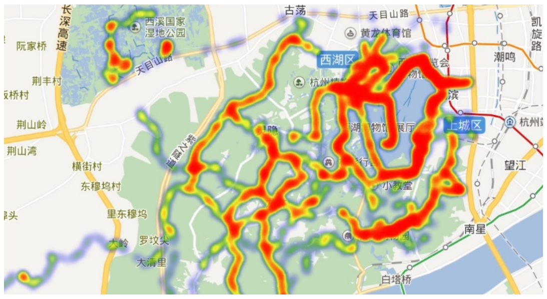


Fig. 15. Heat Map

X. Conclusions and Future Works

For this project, we developed a program based on WebGIS for House Discovery. Completed the system, database, interface, and interface design.

We have successfully implemented basic functions such as user login and registration. At the same time, we use Spring Security's components to implement user authentication system to ensure the user's login authority and security. The user authentication system can also be used to distinguish between ordinary users and administrator roles. The method of defining user roles is safer than the common database field roles.

This project also realizes the function of real-time acquisition of house data. Using Java Springboot components, we have developed a Web Spider that can update house information online every day, so that if the house is sold or the information is no longer public, the user can use our system that obtains up-to-date and useful housing information.

This program can display the specific location of the house on the map according to the acquired house information. This function has also added a number of extensions, such as using bus, school and other surrounding landmark building information to obtain the required house information.

At present, due to time constraints and development feasibility considerations during the project analysis phase, we switched from the python Flask development to Java Springboot and Spring Security in the background development process to implement user authentication. In addition, because the Hibernate component is more suitable for single-table operation , It will encounter many tricky problems when dealing with multiple tables and complex database queries, and it encounters a bottleneck in the process of program implementation. We spent a lot of time debugging and finding solutions. Therefore, this project still has a lot of room for improvement. For example, Spring Security is more suitable for the development of Role-based access system with front-end, admin roles can be used for different users through the system management page The role and permission configuration.

In the future we will continue to polish the existing modules and to develop unfinished functions as we explained above in detail. This project can expand to a fully commercial-capable model that helps people to find their desired house.